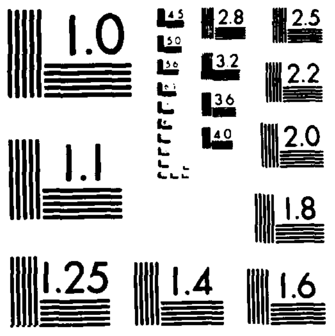
AD-A171 733   A SORTING SYSTEM USING VERY LOW RESOLUTION OPTICAL SENSOR ARRAY IN ROBOT FINGERTIPS(U) CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST R Q YANG ET AL.   1/1

UNCLASSIFIED   MAY 86 CMU-RI-TR-86-10   F/G 13/8   NL


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0

AD-A171 733

86 9 10 067

2

A Sorting System Using Very Low Resolution Optical Sensor Array in Robot Fingertips

R. Q. Yang and M. W. Siegel

CMU-RI-TR-86-10

Shenyang Institute of Automation
Chinese Academy of Sciences
Shenyang, China

Intelligent Sensors Laboratory
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

May 1986

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER CMU-RI-TR-86-10	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Sorting System Using Very Low Resolution Optical Sensor Array in Robot Fingertips		5. TYPE OF REPORT & PERIOD COVERED Interim
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) R. Q. Yang and M. W. Siegel		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Carnegie-Mellon University The Robotics Institute Pittsburgh, PA 15213		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE May 1986
		13. NUMBER OF PAGES 24
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Approved for public release; distribution unlimited

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A low-cost optical sensory sorting system is described. The sensor is directly mounted on robot gripper fingers, a light source on one and a coherent bundle of optical fibers on the opposing one. The optical fibers carry the shadow of a gripped object, as an eight-by-eight pixel array, to detection, multiplexing, discrimination, and computer interface electronics mounted on the robot base. The system uses a microcomputer for several data processing and pattern recognition functions. This discussion covers the design and analysis of the sensor and its optimal array, the hardware, and the parts recognition and control system. System

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

86 9 10 067

(20. cont.)

performance in a demonstration task requiring the acquisition, identification, and sorting of a variety of electronic and mechanical parts is described.

Table of Contents

List of Figures

Abstract

A low-cost optical sensory sorting system is described. The sensor is directly mounted on robot gripper fingers, a light source on one and a coherent bundle of optical fibers on the opposing one. The optical fibers carry the shadow of a gripped object, as an eight-by-eight pixel array, to detection, multiplexing, discrimination, and computer interface electronics mounted on the robot base. The system uses a microcomputer for several data processing and pattern recognition functions. This discussion covers the design and analysis of the sensor and its optimal array, the hardware, and the parts recognition and control system. System performance in a demonstration task requiring the acquisition, identification, and sorting of a variety of electronic and mechanical parts is described.

1. Introduction

Parts often need to be sorted before packing, conveying or mounting, and a variety of sorting systems are in common use in industry. The most common approach is to use a camera for recognizing parts, and a gripper or fingers for picking or mounting. This kind of eye-hand coordination system is highly anthropomorphic and in principle is thus a good prospect for directly replacing human workers. But because of its high cost, bulk, need for extensive computer support, and the slowness of image analysis, many manufacturers are reluctant to consider it when their parts are small, their designs change often, or when they need large numbers of such systems. An inexpensive, simple, robust and flexible sorting system would presumably be welcome by the many factories in one or more of these categories.

We have designed and tested a sensory array, incorporated in the gripper system of a conventional robot, which we think can meet these challenges. It is an optical occlusive system with some vision-like characteristics and some tactile-like characteristics. It has a simple principle of operation, low cost, and potentially high speed. With suitable software, it also can sense slip. Discussion in this paper includes the design and analysis of the transducers and the sensor array, electronics, and parts recognition and control software. Even our simple prototype shows good potentiality for practical applications.

2. Design of the Smart Fingers

The fingertip sensors are based on a simple fiber optic principle [8]. An infra-red light source is built into one finger, and an eight-by-eight square array of optical fibers is built into the opposing finger, as shown in Figure 2-1.[1] A shadow image is transmitted through the fibers to a photo-optical detector array removed from the noisy and cluttered work environment. Very small parts (under 8 mm in their largest dimension) are imaged in one frame, and complete images of larger objects are made in a mosaic of multiple frames, by "feeling" them out along a data-directed path. By these means, an object is gripped by the fingers only at a location determined by the sensors to meet programmed appropriateness criteria. Additional sensors, *e.g.*, proximity switches mounted on the hand, detect unexpected obstacles and command evasive action.

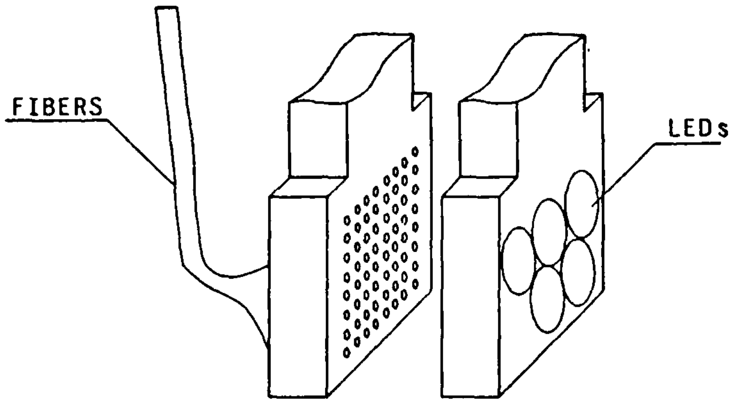


Figure 2-1: Optical illumination and detection arrays

[1] A somewhat similar system employing fiber optics and a linear CCD array is reported by Agrawal and Epstein [1].

2.1. Light Source

The light source is composed of five infra-red light emitting diodes (LEDs), arranged as shown in Figure 2-1. Of course, the ideal light source would emit a parallel beam of uniform spatial intensity. Our less-than-ideal system nevertheless works impressively well, especially after some of the hardware deficiencies are compensated in software.

The angular pattern of each LED has a half cone angle at half intensity of 15°. We have found that 15-30° is a useful compromise between smaller angles, where coverage becomes a problem, and larger angles, where diffuse shadows become a problem.

2.2. Receiver

The eight-by-eight sensor array is composed of sixty-four optical fibers each of diameter 0.5 mm. Each transmits its optical signal to one of sixty-four phototransistors installed behind the base of the robot. Future implementations will be able to make use of integrated photosensitive arrays.

Attenuation by the optical fibers is sufficiently low ($0.5\ \mathrm{dB{\cdot}m^{-1}}$) that the transmission loss is no problem. Phototransistor response times are poor compared with photodiodes, but sensitivity is more useful than speed in our application, and phototransistors are 100 - 500 times more sensitive than photodiodes. With small load resistors ($2\ \mathrm{k\Omega}$), a more than adequate response time is obtained.

2.3. Design and Analysis of the Array Geometry

The resolution of the sensor array is important for the design of the whole system. Resolution requirements are determined by the application, with due consideration to cost, speed, space, weight, etc. In this section we estimate the resolution requirements for detecting and recognizing some small objects. Combining these theoretical considerations with some speculation about the sizes and shapes of objects we might like to sort leads to an appropriate array design.

2.3.1. Pixel Layout

Many pixel layout geometries are in common use; in addition many elementary pixel shapes are possible. In our configuration, we are limited by the optical fibers to circular pixels. These might be arranged in rectangular arrays with various inter-pixel separation, or, in tessellated triangular or hexagonal arrays. We have chosen a rectangular array with inter-pixel separation equal to twice the pixel diameter, shown as "Type B" in Figure 2-2. To give the reader a sense of the considerations in these choices, we will compare the expected performance of this array with that of the one labelled "Type A" in Figure 2-2, where the inter-pixel separation is equal to the pixel diameter.

2.3.2. Critical Detectable Object (CDO)

We assume that image is binary, and all pixels have equal integral and uniform differential sensitivity. The CDO is then defined as the smallest object that switches the state of one pixel. For the system to have the same detection probability for "object" and "background" features of the same area, the signal threshold should be half way between the maximum and minimum intensity levels [4]. Thus

$$\mathrm{Area(CDO)} = 0.5 \times \mathrm{Area(pixel)}$$

The CDO measure is not too useful in practice because it only shows that an object *may* be detected, but it does not guarantee that it *will* be detected. For example, a square may be detected when its side is 0.63 pixel diameters, and an infinite rod may be detected when its width is 0.39 pixel diameters, but the detection probabilities are only 0.013 and 0.0 respectively [4].

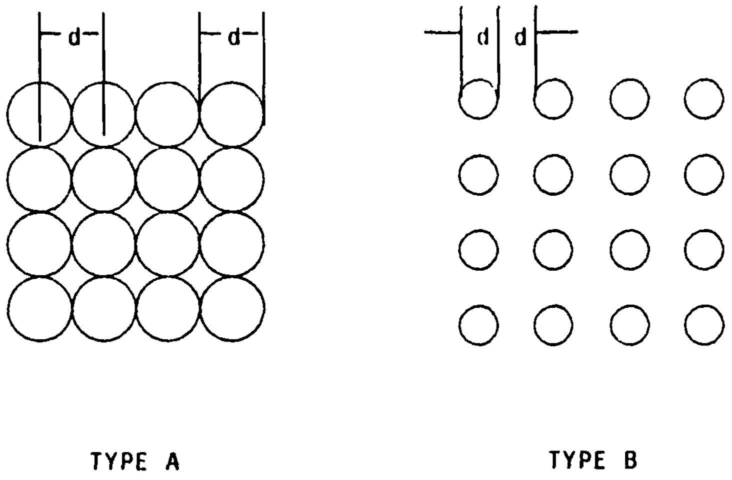


Figure 2-2: Two geometries for circular sensory cell arrays

2.3.3. Minimum Detectable Object (MDO)

The MDO is the smallest object that can always be detected in any position on the sensor array. Although MDO only means that the object can be detected and does not guarantee that it can be recognized, it is nevertheless a practically useful concept.

The MDO size for some generic shapes (discs, squares, triangles) have been analyzed. The worst cases for detection of these objects are shown in Figure 2-3. The results of the analysis are shown in Figure 2-4. In our case (Type B), for example, to guarantee detection of a disc its diameter must be three times the diameter of the fiber. Similarly, to guarantee detection of a square of unit side the diameter of a fiber must be less than 0.42 units. This guarantees detection, but not identification: if a disc and square (or a disc and a hexagon) of the same area are to be differentiated, much higher resolution will in general be required. The most distinguishing feature between a disc and a square is similar to a 90° isosceles triangle, and for a disc and a hexagon it is similar to a 120° isosceles triangle, as shown in Figure 2-5. Simple geometrical considerations based on Figure 2-5 lead to the conclusion that a square and disc can be distinguished when $D > 23.7d$, where D is the disc diameter and d is the pixel diameter. Similarly, a hexagon and a disc can be distinguished when $D > 51.8d$. Figure 2-4 shows some additional examples in graphic form. The last two table entries apply to differentiating a disc and a square of the same area, as discussed.

In every case, noise and mechanical tolerances will ma᾿ ᵓ the real resolution requirement even higher.

3. Hardware

Optical signals from the phototransistors are sent to the computer for data-driven data acquisition, control, and object recognition and sorting. The hardwa͵e configuration to achieve this is shown in Figure 3-1.

In our prototype design, based in part on being able to utilize existing resources, we multiplexed the sixty-four phototransistor signals into a single amplifier, compensating for unit-to-unit variation by individually selecting the phototransistor load resistors, $R_{1,i}$ in Figure 3-2. Resistor R_3 sets the average threshold for creating a binary image. Resistor R_1 influences both the gain of the amplifier and the response time of the phototransistors.

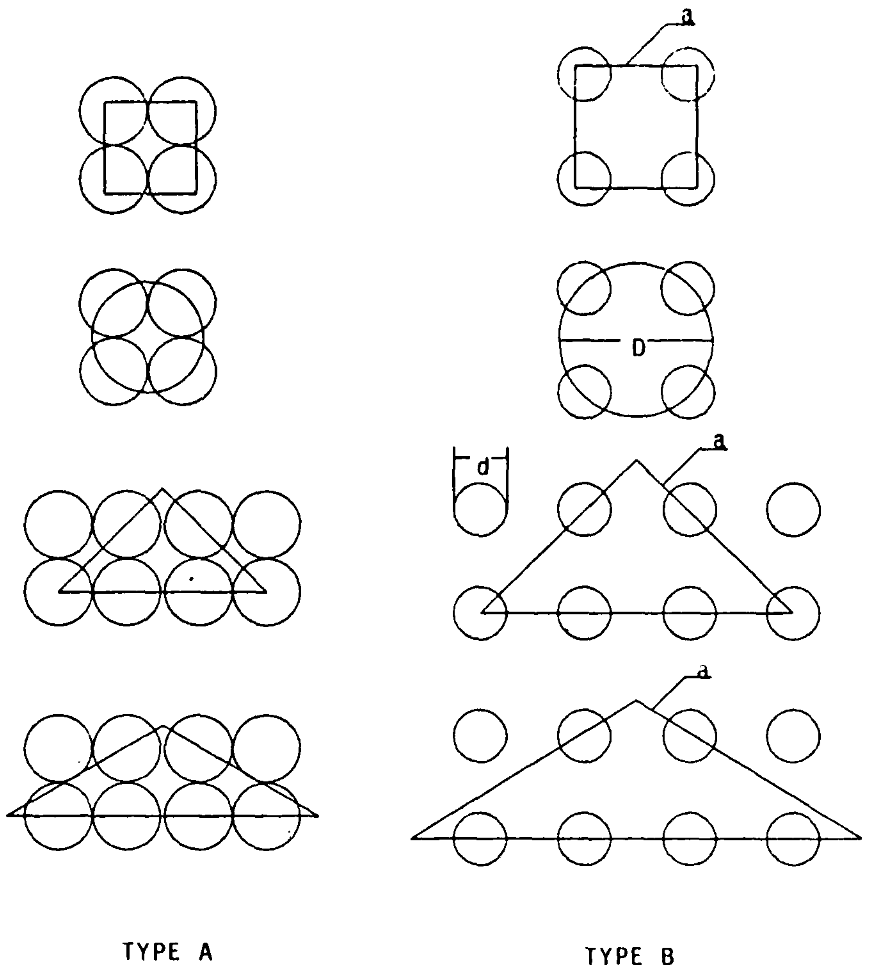


Figure 2-3: Worst cases for object detection

The ADC converts the scaled analog signals to eight-bit digital signals. These are sent to the computer through a bit selectable parallel input/output port. The control signals for the multiplexer and the ADC are output from the same parallel port. The signal of the auxiliary proximity sensor is routed to another parallel port of the computer. The communications link between the computer and robot is through the computer's RS-232c serial port.

4. Object Recognition

The process used to recognize objects can be broken down in to three major steps [3]:

1. Individual pixel data are assembled into an image;

2. Object recognition features are extracted;

3. Detected features are compared with entries in a geometrical or training set data base for object identification.

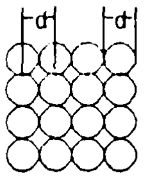
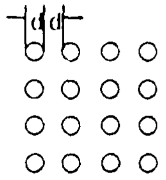
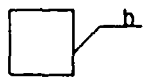
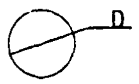
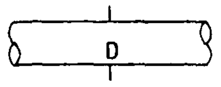
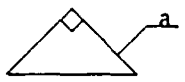
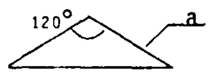
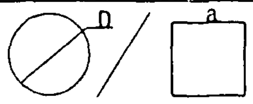
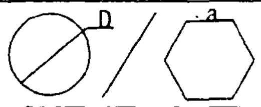
TYPE / OBJECTS		
	b=1.4d	b=2.4d
	D=1.5d	D=3.0d
	D=1.0d	D=2.0d
	a=2.12d	a=4.24d
	a=2.58d	a=5.16d
	or D=11.8d a=10.6d	or D=23.7d a=21.2d
	or D=25.9d a=14.2d	or D=51.8d a=28.3d

Figure 2-4: Minimum detectable objects for "Type A" and "Type B" arrays

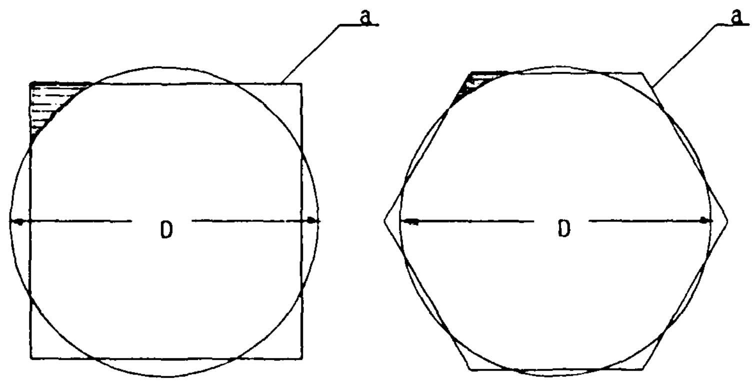


Figure 2-5: Critical features for distinguishing between a disc and a square, and a disc and a hexagon

The number of specific methods available for each step is large. Because our task is to recognize simple objects rapidly, we chose the simplest adequate methods.

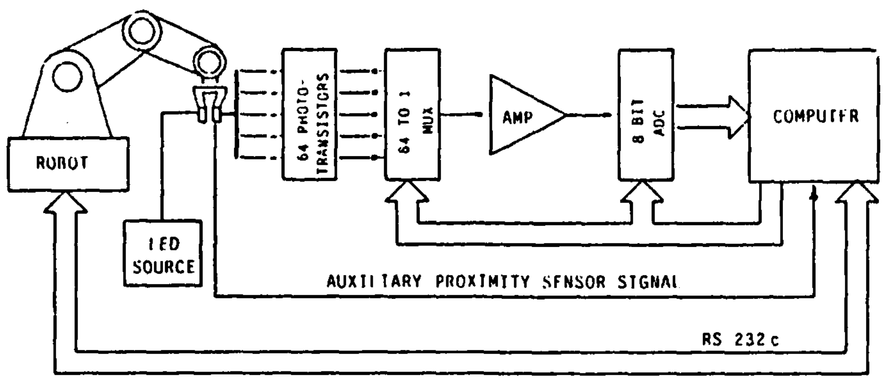


Figure 3-1: Sorting system block diagram

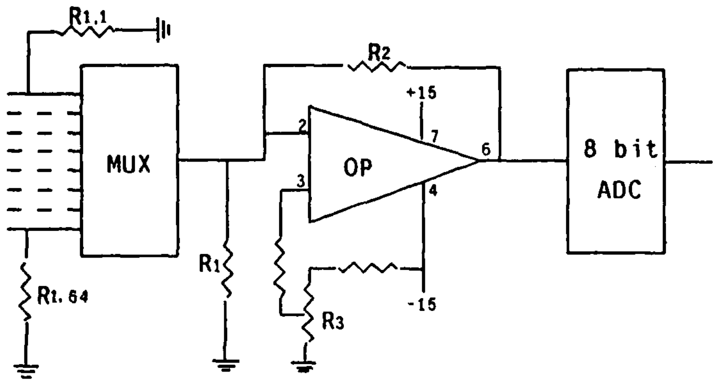


Figure 3-2: Hardware compensation via load resistor array

4.1. Segmentation of the Data

In our application, where gray-scale is irrelevant (because our images are shadows of solid objects), we can segment a picture into "object" and "background" by simply choosing a suitable brightness threshold. We define all pixels whose brightness is below the threshold as "object" and all above the threshold as "background." As mentioned in Section 2.3.3., we can set a threshold half way between the maximum and minimum intensity levels in order to assure that there is parity between figure and background in the image. But in practice, even after hardware compensation by the resistors $R_{1,i}$, the signal due to each unobstructed pixel is different. Thus an additional "soft compensation" table is stored in memory and used to equalize individual pixel sensitivities.

4.2. Feature Extraction and Matching

Very low resolution (VLR) shadow imaging is simple and efficient, but it results in high uncertainty about the actual shape of small complex objects. Also, it is noisy: the apparent locations of the boundaries are very sensitive to slight movement of the object. Thus the boundaries and perimeters of the image alone may not be suitable features for object recognition.

We are investigating two approaches to solving this problem. The first is to take a heuristic approach, in what we believe to be a human-like way, to extract what we think are the essential features of the object independent of minor sensor induced distortions. In our case, we use a program shown as (A) in Figure 4-1 to find the image area (MS), width (IP), length (NM), and existence of enclosed background at the centroid location (IX0,IZ0) as a feature set[2]. This method, while efficient to program and execute, requires inspiration to find an appropriate set of features for the set of objects being recognized. The second approach, which is analytic rather than heuristic, uses normalized quadtree representations for shape matching [2]. This method is universal and convenient, but is costly in program length and execution time.

The former method is used in our demonstration of the prototype sensor. This demonstration discriminates among seven objects (a capacitor, a 20-pin DIP, a 14-pin DIP, three different machine screws, and a nut). Four parameters (area, length, width, and the existence of any enclosed background) are extracted and matched against the stored feature library shown as (B) in Figure 4-1. Three sets of twenty-one trials were run, with each object being presented three times in each trial. The result of the trials shown in Figure 4-2.

5. Control System

A TeachMover [5] robot arm is employed in the demonstration our sorting system.

5.1. Summary of TeachMover Arm Capabilities

The TeachMover robot arm is a microprocessor controlled, six-jointed mechanical arm. The design and performance characteristics of interest to us are:

- five revolution axes, and a gripper;
- electric stepper motors, with open loop control:
  - resolution: 0.011 inch (0.25 mm) maximum on each axis;
  - velocity: 0-7 in-sec^{-1} (0-178 mm-sec^{-1}), with controlled acceleration;
  - load capacity: 16 oz (445 gm) at full extension;
  - gripping force: 3 lbs (13 N) maximum.
- dual RS-232c asynchronous serial communications interfaces;
- typical interface commands; transmitted from the computer to the robot as ASCII text:
  - @CLOSE: close gripper until grip switch is activated;
  - @READ: read values of the internal position registers;
  - @STEP: sets arm speed, moves joints.

[2]The X-axis is radial and Z-axis is vertical, as defined in Figure 5-1

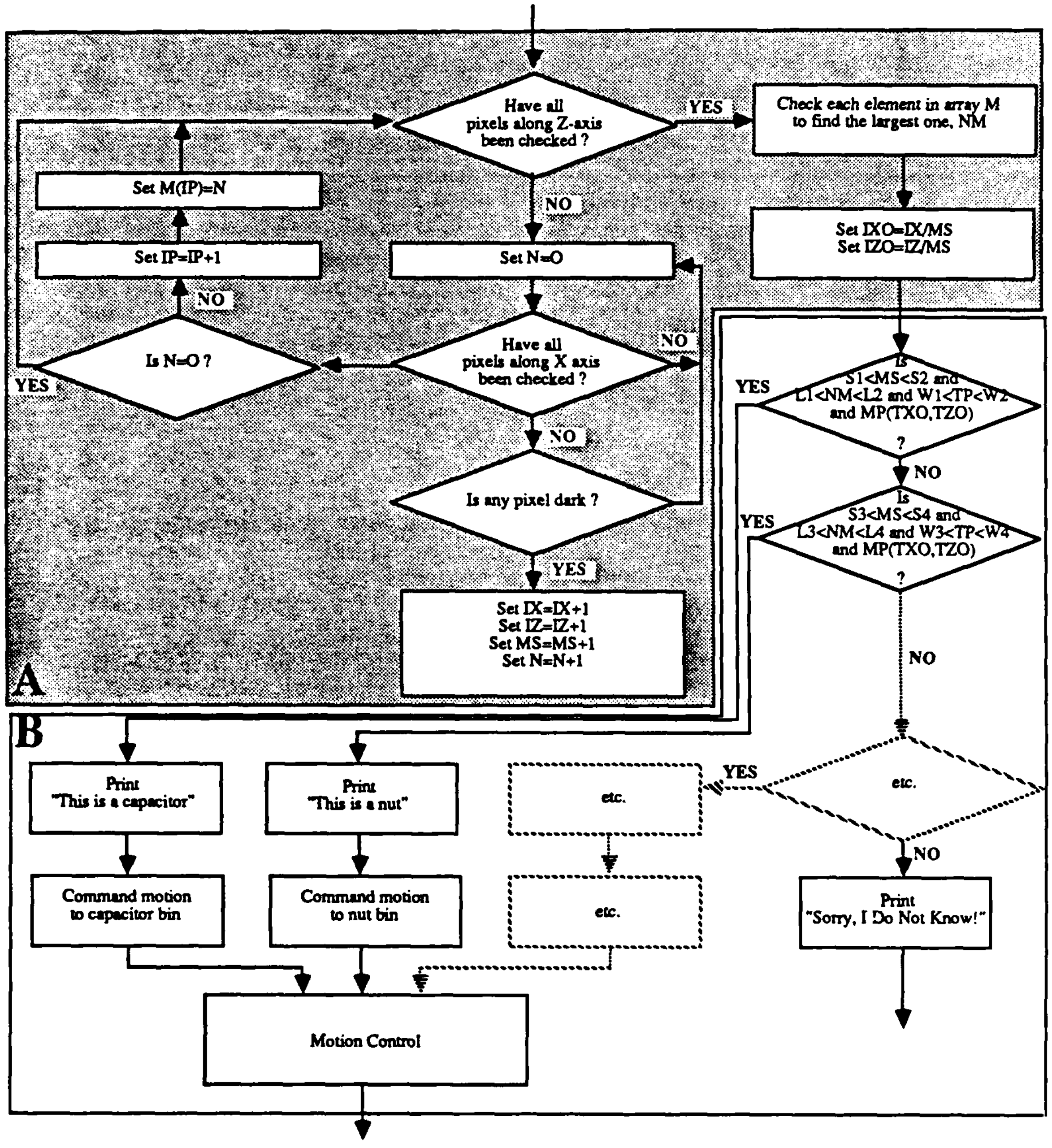

Have all pixels along Z-axis been checked ?
YES
Check each element in array M to find the largest one, NM
NO
Set M(IP)=N
Set IP=IP+1
NO
Set N=O
Set IXO=IX/MS
Set IZO=IZ/MS
Is N=O ?
YES
Have all pixels along X axis been checked ?
NO
NO
Is any pixel dark ?
YES
Set IX=IX+1
Set IZ=IZ+1
Set MS=MS+1
Set N=N+1
A
Is S1<MS<S2 and L1<NM<L2 and W1<TP<W2 and MP(TXO,TZO) ?
YES
NO
Is S3<MS<S4 and L3<NM<L4 and W3<TP<W4 and MP(TXO,TZO) ?
YES
NO
B
Print "This is a capacitor"
Print "This is a nut"
etc.
YES
etc.
NO
Command motion to capacitor bin
Command motion to nut bin
etc.
Print "Sorry, I Do Not Know!"
Motion Control

Figure 4-1: Flowchart for object recognition

5.2. Scanning a Composite Image

The eight-by-eight array, with 1 mm resolution, is too small to image any but the smallest object in a single frame, but a complete image of bigger parts can be obtained by moving the hand. The procedure is as follows:

```
THIS IS A CAPACITOR.    S:L:H=  103  16   8
TRY AGAIN ? (Y/N)Y
THIS IS A 20 PIN CHIP.  S:L:H=  118  26   8
TRY AGAIN ? (Y/N)Y
THIS IA A 14 PIN CHIP.  S:L:H=  115  21   8
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#3-7.   S:L:H=   14   3   7
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#4-20.  S:L:H=   70   8  21
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#5-13.  S:L:H=   67   9  12
TRY AGAIN ? (Y/N)Y
THIS IS A NUT.          S:L:H=  108  15  13
TRY AGAIN ? (Y/N)Y
THIS IS A CAPACITOR.    S:L:H=  109  16   8
TRY AGAIN ? (Y/N)Y
THIS IS A 20 PIN CHIP.  S:L:H=  112  25   8
TRY AGAIN ? (Y/N)Y
THIS IA A 14 PIN CHIP.  S:L:H=  109  22   8
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#3-7.   S:L:H=    9   4   5
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#4-20.  S:L:H=   97   9  24
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#5-13.  S:L:H=   75   9  15
TRY AGAIN ? (Y/N)Y
THIS IS A NUT.          S:L:H=  103  15  13
TRY AGAIN ? (Y/N)Y
THIS IS A CAPACITOR.    S:L:H=  118  16   9
TRY AGAIN ? (Y/N)Y
THIS IS A 20 PIN CHIP.  S:L:H=  121  25   8
TRY AGAIN ? (Y/N)Y
THIS IA A 14 PIN CHIP.  S:L:H=   99  21   8
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#3-7.   S:L:H=   11   4   6
TRY AGAIN ? (Y/N)Y
**SQ**
A.
THIS IS A SCREW-#5-13.  S:L:H=   58   9  14
TRY AGAIN ? (Y/N)Y
THIS IS A NUT.          S:L:H=   98  15  13
TRY AGAIN ? (Y/N)N
STOP
A.


THIS IS A CAPACITOR.    S:L:H=  105  15   8
TRY AGAIN ? (Y/N)Y
THIS IS A 20 PIN CHIP.  S:L:H=  116  25   8
TRY AGAIN ? (Y/N)Y
THIS IA A 14 PIN CHIP.  S:L:H=  113  21   8
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#3-7.   S:L:H=   11   4   6
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#4-20.  S:L:H=   82   8  22
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#5-13.  S:L:H= 62    9  13
TRY AGAIN ? (Y/N)Y
THIS IS A NUT.          S:L:H=  101  15  13
TRY AGAIN ? (Y/N)Y
THIS IS A CAPACITOR.    S:L:H=  112  16   8
TRY AGAIN ? (Y/N)Y
THIS IS A 20 PIN CHIP.  S:L:H=  117  26   8
TRY AGAIN ? (Y/N)Y
THIS IA A 14 PIN CHIP.  S:L:H=  113  21   8
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#3-7.   S:L:H=   10   4   6
TRY AGAIN ? (Y/N)Y
```

```
THIS IS A SCREW-#4-20.  S:L:H=   88   8  23
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW #5-13.  S:L:H=   66   9  14
TRY AGAIN ? (Y/N)Y
THIS IS A NUT.          S:L:H=  109  16  13
TRY AGAIN ? (Y/N)Y
THIS IS A CAPACITOR.    S:L:H=  107  16   8
TRY AGAIN ? (Y/N)Y
THIS IS A 20 PIN CHIP.  S:L:H=  115  25   8
TRY AGAIN ? (Y/N)Y
**SQ**
A.
THIS IS A SCREW-#3-7.   S:L:H=   11   4   6
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#4-20.  S:L:H=   82   8  22
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#5-13.  S:L:H=   65   9  13
TRY AGAIN ? (Y/N)Y
THIS IS A NUT.          S:L:H=  102  15  13
TRY AGAIN ? (Y/N)N
STOP
A.


THIS IS A CAPACITOR.    S:L:H=  105  15   8
TRY AGAIN ? (Y/N)Y
THIS IS A 20 PIN CHIP.  S:L:H=  116  26   8
TRY AGAIN ? (Y/N)Y
THIS IA A 14 PIN CHIP.  S:L:H=  110  21   8
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#3-7.   S:L:H=   13   4   6
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#4-20.  S:L:H=   92   9  22
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#5-13.  S:L:H=   61   9  13
TRY AGAIN ? (Y/N)Y
THIS IS A NUT.          S:L:H=  107  15  13
TRY AGAIN ? (Y/N)Y
THIS IS A CAPACITOR.    S:L:H=  109  16   8
TRY AGAIN ? (Y/N)Y
THIS IS A 20 PIN CHIP.  S:L:H=  114  25   8
TRY AGAIN ? (Y/N)Y
THIS IA A 14 PIN CHIP.  S:L:H=  108  21   8
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#3-7.   S:L:H=   10   3   5
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#4-20.  S:L:H=   89   8  22
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#5-13.  S:L:H=   64   9  13
TRY AGAIN ? (Y/N)Y
THIS IS A NUT.          S:L:H=  104  15  13
TRY AGAIN ? (Y/N)Y
THIS IS A CAPACITOR.    S:L:H=  114  16   9
TRY AGAIN ? (Y/N)Y
THIS IS A 20 PIN CHIP.  S:L:H=  112  25   8
TRY AGAIN ? (Y/N)Y
THIS IA A 14 PIN CHIP.  S:L:H=  113  22   8
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#3-7.   S:L:H=   11   4   6
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#4-20.  S:L:H=   83   8  22
TRY AGAIN ? (Y/N)Y
THIS IS A SCREW-#5-13.  S:L:H=   70   9  14
TRY AGAIN ? (Y/N)Y
THIS IS A NUT.          S:L:H=  106  15  13
TRY AGAIN ? (Y/N)N
STOP
A.
```

Figure 4-2: Trial results (enclosed background check not printed out)

1. Check whether the image is complete, without regard to recognition; if incomplete, generate a strategy for moving the hand;

2. Read and calculate the old joint angles (B_{20} and B_{30} in Figure 5-1);

3. Evaluate the required new joint angles (B_2 and B_3 in Figure 5-1) to reach the desired new position;

4. Check the composite image of the part again; if it is complete, command the robot to move the hand to the middle of the part (or other suitable gripping point); if not, go back to step(2).

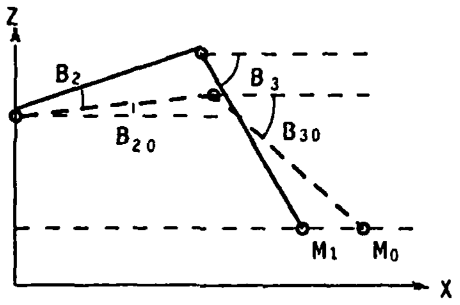


Figure 5-1: Simplified three-link articulated manipulator

5.3. Motion Control

Motion control is simple, because the TeachMover arm uses stepper motors, and open-loop control is usually adequate:

1. The computer issues a robot motion command, e.g. @CLOSE, @STEP.

2. After executing the motion, the TeachMover arm sends an acknowledgement signal to the computer, signalling that it is ready for next command;

3. Repeat steps (1) and steps (2) until task execution is finished.

If an obstacle is encountered by the proximity sensor, an interrupt will be generated, causing the computer to command an avoidance maneuver.

6. Conclusion

A simple, low cost small parts sorter based on "smart fingers" has been developed and demonstrated. The system uses an eight-by-eight optical fiber array with 1 mm resolution. It can recognize and sort parts larger than the array itself by scanning and assembling a composite image.

Because the sensor array is located in the fingers, the information content per bit is very high, and a very small number of bits is usually sufficient for part recognition. The data-base of parts can similarly be stored very compactly, and very efficient identification algorithms can be executed in a limited capability microcomputer. The heuristic approach to extracting essential features for object classification is attractive and efficient. The ability to use auxiliary sensors to interrupt and affect motion control is an especially flexible way to deal with obstacles along the intended trajectory.

There are, of course, limitations to this system. For example, it is difficult to distinguish between similar small complex parts using low resolution, and the parts being recognized must be located *a priori* within a fairly small work volume to avoid extensive blind search. Also, we have not attempted to address the problem of bin-picking or overlapping parts: our system presents the parts, on a simulated convey or belt, one at a time.

The theoretical analysis of the resolution of pixel arrays provides a foundation for the future design of the special purpose sensor arrays. The design of optimal array geometries for specific applications will bring significant advantages with respect to maximizing information density, minimizing cost, and simplifying hardware and software.

7. References

1. A. Agrawal. Robot Eye-In-Hand Using Fiber Optics. Intelligent Robots, Intelligent Robots: Third International Conference on Robot Vision and Sensory Controls, November, 1983, pp. 48. Volume 449.

2. C. H. Chien, and J. K. Agarwal. "A Normalized Quadtree Representation". *Computer Vision, Graphics, and Image Processing 26* (1984), 331-46.

3. J. P. Christ, and A. Sanderson, A Prototype Tactile Sensor Array. Carnegie-Mellon University, September , 1982. A technical report of the C-MU Robotics Institute.

4. M. H. Lee and F. S. Shahabi. "Very Low Resolution Vision Sensors Offer Gains in Cost and Speed". *Sensor Review* (October 1982).

5. *TeachMover User Reference Manual.* Microbot, Inc., 1982.

6. R. P. Paul. *Robot Manipulators.* The MIT Press, 1981.

7. W. E. Snyder. *Industrial Robots: Computer Interfacing and Control.* Prentice-Hall, Inc., 1985.

8. P. M. Taylor, G. E. Taylor, D. R. Kemp, J. Stein, and A. Pugh, "Sensory Gripping System: The Software and Hardware Aspects ". *Sensor Review* (October 1981).

APPENDIX

```
C       MAIN ROUTINE FOR SORTER
        DIMENSION MP(40,40),MD(8,8),M(40),NAM(6)
        COMMON /CR/VE,HO/PF/NI,NJ,MP,MD/RU/JJ,NAM,MS,NM,IP,IB,IC
        CALL OUT(80,144)
        WRITE(5,6)
6       FORMAT(1X,'DISTANCE(5000--15000):')
        READ(5,7) IDIS
7       FORMAT(I6)
        WRITE(5,8)
8       FORMAT(1X,'INTERVAL(500--1500):')
        READ(5,7) INTV
        WRITE(5,9)
9       FORMAT(1X,'CALIBRATE ? (Y/N)')
        READ(5,10) IC
10      FORMAT(A1)
        IF(IC.EQ.'N'.OR.IC.EQ.'n') GO TO 15
        CALL INIT
11      WRITE(5,12)
12      FORMAT(1X,'READY ? (Y/N)')
        READ(5,13) ID
13      FORMAT(A1)
        IF(ID.EQ.'N'.OR.ID.EQ.'n') GO TO 11
15      DO 24 I=1,40
        DO 26 J=1,40
        MP(I,J)='o'
26      CONTINUE
24      CONTINUE
        IX=0
        JY=0
        MS=0
        NM=0
        IP=0
        NI=0
        NJ=0
        NJ1=0
25      IP1=0
        I=40

C       MOVE HANDS UP TO AVOID MISSLEADING BY ALUMINUM BLOCK
        CALL PEK
28      IP1=IP1+1
        DO 30 J=1,8
        IF(MP(I,J).EQ.'o') GO TO 50
30      CONTINUE
        I=I-1
        IF(IP1.GE.8) GO TO 50
        GO TO 28
50      VE=FLOAT(IP1)-0.5
        HO=0.
        CALL CRUN
```

```
C       MOVE HANDS LATERRALLY TO ONE EDGE OF IMAGE
150     CALL PEK
        DO 155 J=1,8
        DO 158 I=33,40
        JP=J
        IF(MP(I,J).EQ.'*') GO TO 160
158     CONTINUE
155     CONTINUE
        WRITE(5,790) ((MP(I,J),J=1,8),I=33,40)
790     FORMAT(1X,8A2)
        GO TO 450
160     IF(JP.EQ.1) GO TO 165
        VE=0.
        HO=FLOAT(JP)-1.5
        CALL CRUN
        GOTO 65
165     HO=-4.0
        VE=0.
        CALL CRUN
        GO TO 150

C       SCAN COMPOSITE IMAGE
65      CALL PEK
        J=8*(NJ+1)
        K=33-NI*8
        K1=K+7
        DO 70 I=K,K1
        IF(MP(I,J).EQ.'*') GO TO 80
70      CONTINUE
        I=K
        K=J
95      DO 75 J=1,K
        IF(MP(I,J).EQ.'*') GO TO 85
75      CONTINUE
        GO TO 100
80      NJ=NJ+1
        IF(NJ.GT.NJ1)   NJ1=NJ
        IF(NJ.EQ.5) GO TO 300
        HO=8.0
        VE=0.0
        CALL CRUN
        GO TO 65
85      NI=NI+1
        IF(NI.EQ.5) GO TO 300
        HO=0.0
        VE=8.0
        CALL CRUN
        IF(NJ.GT.0) GO TO 90
        GO TO 65
90      CALL PEK
        HO=-8.0
        VE=0.0
        CALL CRUN
        NJ=NJ-1
        IF(NJ.GT.0) GO TO 90
        CALL PEK
        I=33-NI*8
        K=8*(NJ1+1)
        GO TO 95
300     WRITE(5,305)
305     FORMAT(1X,'TOO BIG!')

C       MOVE HANDS TO THE MIDDLE OF THE OBJECT
100     K1=8*(NJ1+1)
        K2=33-NI*8
        NJ2=NJ1+1
500     IF(NJ1.EQ.0.AND.NI.EQ.0) GO TO 510
        VE=-FLOAT(NI*8)+1.5
        HO=FLOAT(NJ1*4)
        IF(NJ1.EQ.0) HO=0.0
        IF(NJ.GT.0) HO=-(HO-(NJ1-NJ)*8)
        CALL CRUN

C       EXTRACT FEATURES OF OBJECT
510     DO 130 I=K2,40
        N=0
        DO 140 J=1,K1
        IF(MP(I,J).NE.'*') GO TO 140
        IX=IX+I
        JY=JY+J
        MS=MS+1
        N=N+1
140     CONTINUE
        IF(N.EQ.0) GO TO 130
        IP=IP+1
        M(IP)=N
130     CONTINUE
        DO 132 J=1,K1
        DO 134 I=K2,40
        IF(MP(I,J).EQ.'*') GO TO 136
134     CONTINUE
        GO TO 132
136     NM=NM+1
132     CONTINUE
        IX0=IX/MS
        JY0=JY/MS

C       OBJECT IDENTIFICATION
        IF(MS.GE.70.AND.MS.LE.140.AND.NM.GE.12.AND.NM.LE.17.AND.
     1  IP.GE. 7.AND.IP.LE.15 .AND.MP(IX0,JY0).EQ.'*') GO TO 210
        IF(MS.GE. 80.AND.MS.LE.150.AND.NM.GE.23.AND.NM.LE.28.AND.
     1     IP.GE.  6.AND.IP.LE.  9) GO TO 220
        IF(MS.GE. 50.AND.MS.LE.120.AND.NM.GE.18.AND.NM.LE.22.AND.
     1     IP.GE.  6.AND.IP.LE.  9) GO TO 230
        IF(MS.GE.  2.AND.MS.LE. 30.AND.NM.GE. 1.AND.NM.LE. 8.AND.
     1     IP.GE.  1.AND.IP.LE.  8) GO TO 240
        IF(MS.GE. 50.AND.MS.LE.110.AND.NM.GE. 5.AND.NM.LE.10.AND.
     1     IP.GE. 19.AND.IP.LE. 26) GO TO 250
        IF(MS.GE. 30.AND.MS.LE.100.AND.NM.GE. 5.AND.NM.LE.11.AND.
     1     IP.GE.  6.AND.IP.LE. 18) GO TO 260
        IF(MS.GE. 60.AND.MS.LE.120.AND.NM.GE.12.AND.NM.LE.18.AND.
     1     IP.GE. 11.AND.IP.LE. 15) GO TO 270
```

```
      GO TO 460
210   JJ=IDIS+1*INTV
      NAM(1)='CA'
      NAM(2)='PA'
      NAM(3)='CI'
      NAM(4)='TO'
      NAM(5)='R.'
      NAM(6)='  '
      IC=51
      GO TO 420
220   JJ=IDIS+2*INTV
      NAM(1)='20'
      NAM(2)=' P'
      NAM(3)='IN'
      NAM(4)=' C'
      NAM(5)='HI'
      NAM(6)='P.'
      IC=48
      GO TO 420
230   JJ=IDIS+3*INTV
      NAM(1)='14'
      NAM(2)=' P'
      NAM(3)='IN'
      NAM(4)=' C'
      NAM(5)='HI'
      NAM(6)='P.'
      IC=48
      GO TO 420
240   JJ=IDIS+4*INTV
      NAM(1)='TH'
      NAM(2)='RE'
      NAM(3)='AD'
      NAM(4)='--'
      NAM(5)='7.'
      NAM(6)='  '
      IC=53
      GO TO 420
250   JJ=IDIS+5*INTV
      NAM(1)='TH'
      NAM(2)='RE'
      NAM(3)='AD'
      NAM(4)='--'
      NAM(5)='20'
      NAM(6)='. '
      IC=50
      GO TO 420
260   JJ=IDIS+6*INTV
      NAM(1)='TH'
      NAM(2)='RE'
      NAM(3)='AD'
      NAM(4)='--'
      NAM(5)='13'
      NAM(6)='. '
      IC=49
      GO TO 420
270   JJ=IDIS+7*INTV
```

```
        NAM(1)='NU'
        NAM(2)='T.'
        NAM(3)='  '
        NAM(4)='  '
        NAM(5)='  '
        NAM(6)='  '
        IC=49
        GO TO 420
460     WRITE(5,465) MS,NM,IP
465     FORMAT(1X,3I6/1X,'SORRY, I DO NOT KNOW !')
        GO TO 430
420     CALL SSRUN
430     VE=-3.5
        HO=-FLOAT(NJ1*4)
        CALL CRUN
        GO TO 600
450     WRITE(5,455)
455     FORMAT(1X,'NOTHING!')
600     WRITE(5,470)
470     FORMAT(1X,'TRY AGAIN ? (Y/N)')
        READ(5,475) IB
475     FORMAT(A1)
480     IF(IB.EQ.'N'.OR.IB.EQ.'n') GO TO 700
        GO TO 15
700     STOP
        END



C       SUBROUTINE FOR INITIALIZATION OF ROBOT ARM
        SUBROUTINE INIT
        DIMENSION IRD(6),ID(30),ISTP(9),ISTSH(5),ISTEL(12),
     1  ISTO(2),IBAK(15),IH(17),ISTP1(9),IRSET(7),IRDY(16),
     2  ISH(5),IBR(5),IBL(3),IUP(5)
        DATA IRD/64,82,69,65,68,13/,ISTP/64,83,84,69,80,49,49,49,
     1  44/,ISTSH/44,52,48,48,13/,ISTEL/44,44,51,48,48,44,44,44,
     2  51,48,48,13/,ISTO/48,13/,IBAK/44,45,52,48,44,45,54,48,44,
     3  44,44,45,54,48,13/,IH/44,54,56,50,44,45,57,57,52,44,44,44
     4  ,45,57,57,52,13/,ISTP1/64,83,84,69,80,52,48,48,44/,IRSET/
     5  64,82,69,83,69,84,13/,IRDY/44,45,49,51,50,44,52,49,51,44,
     6  44,44,52,49,51,13/,ISH/44,48,50,50,13/,IBR/45,57,57,57,13
     7  /,IBL/53,48,13/,IUP/44,45,53,48,13/

        IREP=0
        M1=0
        M2=0
        CALL SOUT(ISTP,9)
        CALL SOUT(ISTSH,5)
215     N1=INP(100)
        IF (N1.EQ.0) GO TO 215
        M1=M1+1
        IF(M1.LE.100) GO TO 215
        CALL SOUT(ISTP,9)
        CALL SOUT(ISTO,2)
        CALL REIN
        CALL REIN
        CALL SOUT(ISTP,9)
        CALL SOUT(IBR,5)
220     N2=INP(100)
        IF (N2.EQ.64) GO TO 220
        M2=M2+1
        IF (M2.LE.50) GO TO 220
        CALL SOUT(ISTP,9)
        CALL SOUT(ISTO,2)
        CALL REIN
        CALL REIN
        CALL SOUT(ISTP,9)
        CALL SOUT(IBL,3)
        CALL REIN
        CALL SOUT(ISTP,9)
        CALL SOUT(IUP,5)
        CALL REIN
50      IREP=IREP+1
        M1=0
        M2=0
        CALL SOUT(ISTP,9)
        CALL SOUT(ISTSH,5)
217     N1=INP(100)
        IF(N1.EQ.0) GO TO 217
        M1=M1+1
        IF(M1.LE.1000) GO TO 217
        CALL SOUT(ISTP,9)
        CALL SOUT(ISTO,2)
        CALL REIN
        CALL REIN
        CALL SOUT(ISTP,9)
        CALL SOUT(ISTEL,12)
222     N2=INP(100)
        IF(N2.EQ.64) GO TO 222
        M2=M2+1
        IF(M2.LE.50) GO TO 222
        CALL SOUT(ISTP,9)
        CALL SOUT(ISTO,2)
        CALL REIN
        CALL REIN

        CALL SOUT(IRD,6)
        DO 70 I=1,30
        ID(I)=INP(81)
        DO 80 J=1,13
        G=2*6
80      CONTINUE
70      CONTINUE
        IF(IREP.EQ.3) GO TO 250
        CALL SOUT(ISTP,9)
        CALL SOUT(IBAK,15)
        CALL REIN
        GO TO 50

250     CALL SOUT(ISTP1,9)
```

```
      CALL SOUT(IBAK,15)
      CALL REIN
      CALL SOUT(ISTP1,9)
      CALL SOUT(IH,17)
      CALL REIN
      CALL SOUT(IRSET,7)
      CALL REIN
      CALL SOUT(ISTP1,9)
      CALL SOUT(IRDY,16)
      CALL REIN
      RETURN
      END



C     SUBROUTINE TO BINARY IMAGE
      SUBROUTINE PEK
      INTEGER H,P
      DIMENSION MD(8,8),MP(40,40)
      COMMON /PE/NI,NJ,MP,MD

      DO 10 I=1,8
      DO 20 J=1,8
      N=8*(I-1)+J-1
      CALL OUT(84,N)
      N1=N+64
      DO 30 P=1,6
      F=2*6
30    CONTINUE
      CALL OUT(84,N1)
      CALL OUT(84,N)
      MD(I,J)=INP(84)
      IF(MD(I,J).LT.0) MD(I,J)=256+MD(I,J)
      K=I+32-NI*8
      H=J+NJ*8
      MP(K,H)='o'
      MEN=120
      IF(I.EQ.1.AND.J.EQ.1.OR.I.EQ.1.AND.J.EQ.2) MEN=13
      IF(I.EQ.3.AND.J.EQ.8.OR.I.EQ.1.AND.J.EQ.3.OR.
     1   I.EQ.8.AND.J.EQ.4.OR.I.EQ.8.AND.J.EQ.7) MEN=35
      IF(I.EQ.2.AND.J.EQ.7.OR.I.EQ.7.AND.J.EQ.8.OR.
     1   I.EQ.4.AND.J.EQ.2.OR.I.EQ.3.AND.J.EQ.7) MEN=50
      IF(I.EQ.7.AND.J.EQ.2.OR.I.EQ.2.AND.J.EQ.8.OR.
     1   I.EQ.4.AND.J.EQ.8.OR.I.EQ.8.AND.J.EQ.1.OR.
     2   I.EQ.4.AND.J.EQ.7.OR.I.EQ.2.AND.J.EQ.2.OR.
     3   I.EQ.1.AND.J.EQ.4.OR.I.EQ.2.AND.J.EQ.3) MEN=75
      IF(I.EQ.2.AND.J.EQ.5.OR.I.EQ.8.AND.J.EQ.6) MEN=85
      IF(MD(I,J).LT.MEN) MP(K,H)='*'
20    CONTINUE
10    CONTINUE
      RETURN
      END
```

```
C       SUBROUTINE FOR HANDSHAKING
        SUBROUTINE REIN
        DO 10 J=1,16
        DO 20 I=1,32222
        N1=INP(81)
        IF(N1.EQ.49) GO TO 60
        IF(N1.EQ.48) GO TO 30
20      CONTINUE
10      CONTINUE
30      WRITE(5,40)
40      FORMAT(1X,'WRONG!!')
60      RETURN
        END



C       SUBROUTINE TO MOVE HANDS ACCORDING VE(VERTICAL) AND HO(HORIZONTAL)
        SUBROUTINE CRUN
        DIMENSION IRD(6),ID(40),ND(8),ISTP(9),NB(4),IS(18)
        COMMON /CR/VE,HO
        DATA IRD/64,82,69,65,68,13/,ISTP/64,83,84,69,80,50,50,48,44/,
     1  IS(1),IS(6),IS(11),IS(12),IS(13),IS(18)/5*44,13/
        CALL SOUT(IRD,6)
        DO 70 I=1,40
        ID(I)=INP(81)
        DO 80 J=1,14
        F=2*6
80      CONTINUE
70      CONTINUE
        IF (ID(2).EQ.48) GO TO 500
        NK2=0
        NK4=0
        NK3=1
        DO 90 I=4,36
        IF (ID(I)-45) 230,210,200
200     NK4=NK4+1
        NB(NK4)=ID(I)-48
        GO TO 90
210     NK3=-1
        GO TO 90
230     NK2=NK2+1
        GO TO (205,206,207,208),NK4
205     ND(NK2)=NK3*NB(1)
        GO TO 240
206     ND(NK2)=NK3*(NB(1)*10+NB(2))
        GO TO 240
207     ND(NK2)=NK3*(NB(1)*100+NB(2)*10+NB(3))
        GO TO 240
208     ND(NK2)=NK3*(NB(1)*1000+NB(2)*100+NB(3)*10+NB(4))
240     IF (NK2.EQ.6) GO TO 250
```

```
          NK3=1
          NK4=0
    S0    CONTINUE
    250   D2=FLOAT(ND(2))*3.1416/3536.0
          D3=FLOAT(ND(3))*3.1416/2079.0
          A1=SIN(D2)+SIN(D3)-VE/177.8
          B1=COS(D2)+COS(D3)-HO/177.8
          C1=2.0*A1*B1/(B1*B1-A1*A1)
          C1=ATAN(C1)
          C3=4.0-(A1*A1+B1*B1-2.0)**2
          C2=(A1*A1+B1*B1-2.0)/SQRT(C3)
          C2=2.0*ATAN(1.0)-ATAN(C2)
          E1=(C1+C2)*1039.0/3.1416
          NE1=INT(E1+0.5)-ND(3)
          E2=(C1-C2)*1768.0/3.1416
          NE2=INT(E2+0.5)-ND(2)
          IF(NE2.LT.0) GO TO 345
          IS(2)=48
          GO TO 346
    345   IS(2)=45
    346   IF(NE1.LT.0) GO TO 348
          IS(7)=48
          IS(14)=48
          GO TO 349
    348   IS(7)=45
          IS(14)=45
    349   NE1=IABS(NE1)
          NE2=IABS(NE2)
          IS(3)=INT(FLOAT(NE2/100))+48
          IS(8)=INT(FLOAT(NE1/100))+48
          IS(15)=IS(8)
          IS(4)=INT(FLOAT(NE2/10))-(IS(3)-48)*10+48
          IS(9)=INT(FLOAT(NE1/10))-(IS(8)-48)*10+48
          IS(16)=IS(9)
          IS(5)=INT(FLOAT(NE2))-(IS(3)-48)*100-(IS(4)-48)*10+48
          IS(10)=INT(FLOAT(NE1))-(IS(8)-48)*100-(IS(9)-48)*10+48
          IS(17)=IS(10)
          CALL SOUT(ISTP,9)
          CALL SOUT(IS,18)
          CALL REIN
          GO TO 510
    500   WRITE(5,110)
    110   FORMAT(1X,'WRONG!')
    510   RETURN
          END


    C     SUBROUTINE TO COMMAND ROBOT
          SUBROUTINE SOUT(IO,NDIM)
          DIMENSION IO(NDIM)
          DO 10 I=1,NDIM
          CALL OUT(81,IO(I))
          DO 15 J=1,16
```

```
         F=2*6
15       CONTINUE
10       CONTINUE
         RETURN
         END




C        SUBROUTINE FOR MOTION CONTROL
         SUBROUTINE SSRUN
         DIMENSION ICLS(7),ISTP(9),IST10(18),IST1(18),IST2(5),IST3(15)
      1  ,IST4(9),IST6(6),IST20(6),IST0(2),NAM(6)
         COMMON /RU/JJ,NAM,MS,NM,IP,IB,IC
         DATA ICLS/64,67,76,79,83,69,13/,ISTP/64,83,84,69,80,50,50,48,
      1  44/,IST2/50,53,48,48,13/,IST1/44,45,51,48,48,44,45,49,48,48,
      2  44,44,44,45,49,48,48,13/,IST3/44,49,48,48,44,49,48,48,44,44,
      3  44,49,48,48,13/,IST4/44,44,44,44,44,50,48,48,13/,IST6/45,50,
      4  53,48,48,13/,IST10/44,45,48,50,48,44,45,48,50,48,44,44,44,45,
      5  48,50,48,13/,IST20/45,48,48,48,52,13/,IST0/48,13/
         N1=0
         N4=0
         N6=0
         N8=0
         IST4(7)=IC
         CALL SOUT(ICLS,7)
         CALL REIN
         CALL SOUT(ISTP,9)
         CALL SOUT(IST1,18)
         CALL REIN
210      CALL SOUT(ISTP,9)
         CALL SOUT(IST2,5)
215      N1=N1+1
         F=50**3
         N2=INP(100)
         IF(N1.GE.JJ) GO TO 250
         IF(N2.EQ.0) GO TO 215
         CALL SOUT(ISTP,9)
         CALL SOUT(IST0,2)
         CALL REIN
         CALL REIN
         N8=N8+1
         JJ=JJ+275
220      CALL SOUT(ISTP,9)
         CALL SOUT(IST10,18)
         CALL REIN
         N4=N4+1
         N2=INP(100)
         IF(N2.NE.0) GO TO 220
230      CALL SOUT(ISTP,9)
         CALL SOUT(IST10,18)
         CALL REIN
         N6=N6+1
         IF(N6.EQ.2.OR.N6.EQ.4.OR.N6.GE.6) GO TO 210
      GO TO 230
250   CALL SOUT(ISTP,9)
      CALL SOUT(IST0,2)
      CALL REIN
      CALL REIN
      IS1=(N4+N6)*2
      IS4=IS1/10
      IS2=49+IS4
      IS3=48+IS1-IS4*10
      IST3(2)=IS2
      IST3(3)=IS3
      IST3(6)=IS2
      IST3(7)=IS3
      IST3(12)=IS2
      IST3(13)=IS3
      CALL SOUT(ISTP,9)
      CALL SOUT(IST3,15)
      CALL REIN
      WRITE(5,260)(NAM(I),I=1,6),MS,NM,IP
260   FORMAT(1X,'THIS IS A ',6A2,'  S:L:H=',3I6)
      CALL SOUT(ISTP,9)
      CALL SOUT(IST4,9)
      CALL REIN
      IST1(3)=IS2
      IST1(4)=IS3
      IST1(8)=IS2
      IST1(9)=IS3
      IST1(15)=IS2
      IST1(16)=IS3
      CALL SOUT(ISTP,9)
      CALL SOUT(IST1,18)
      CALL REIN
      N1=0
      JJ=JJ-N8*275
      IST3(2)=IS2+2
      CALL SOUT(ISTP,9)
      CALL SOUT(IST6,6)
270   N1=N1+1
      F=50**3
      N2=INP(100)
      IF(N1.GE.JJ) GO TO 280
      IF(N2.EQ.0) GO TO 270
      IF(N2.NE.0) GO TO 270
280   CALL SOUT(ISTP,9)
      CALL SOUT(IST0,2)
      CALL REIN
      CALL REIN
      CALL SOUT(ISTP,9)
      CALL SOUT(IST3,15)
      CALL REIN
      IST3(2)=49
      IST3(3)=48
      IST3(6)=49
      IST3(7)=48
      IST3(12)=49
      IST3(13)=48
```

```
        IST1(3)=51
        IST1(4)=48
        IST1(8)=49
        IST1(9)=48
        IST1(15)=49
        IST1(16)=48
        WRITE(5,290)
290     FORMAT(1X,'                              ')
        RETURN
        END
```

END

10-86

DTIC